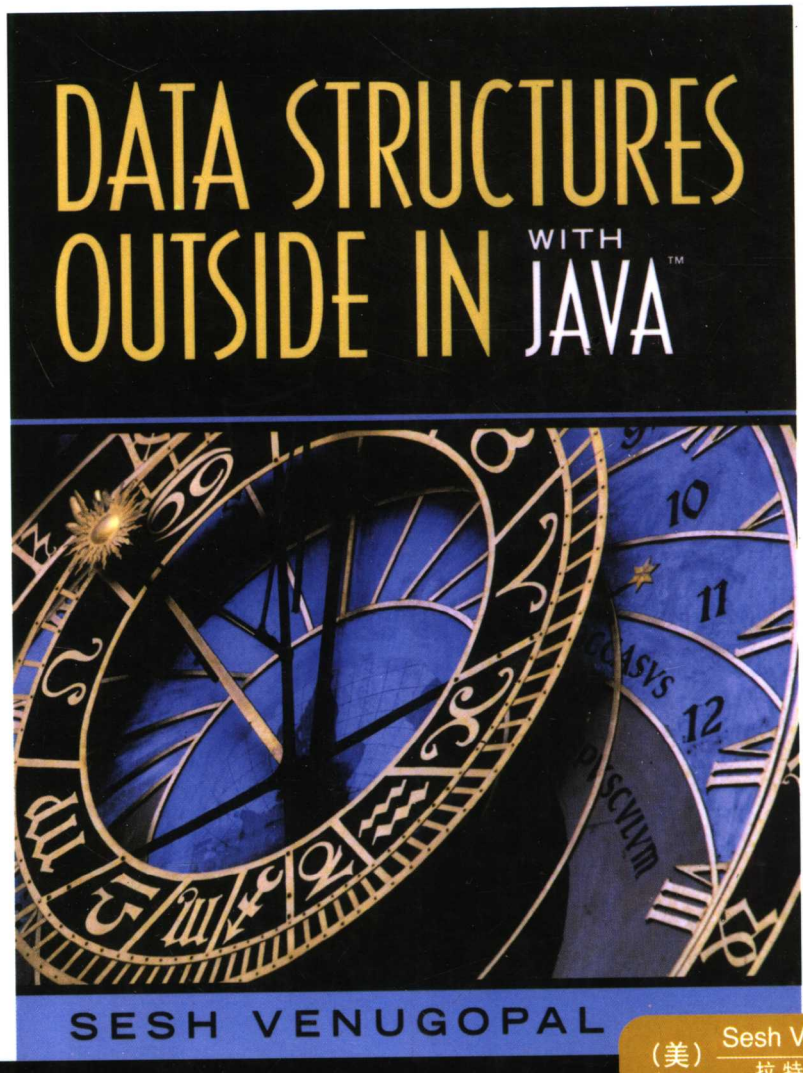
HZ BOOKS 华章教育
经典原版书库
PRENTICE HALL

数据结构

从应用到实现(Java版)

(英文版)



(美) Sesh Venugopal 著
拉特格大学

机械工业出版社
China Machine Press

经 典 原 版 书 库

数据结构

从应用到实现(Java版)

(英文版)

Data Structures Outside In with Java

(美) Sesh Venugopal 著
拉特格大学

机械工业出版社
China Machine Press

本书版权登记号：图字：01-2007-1843

图书在版编目（CIP）数据

数据结构：从应用到实现（Java版）（英文版）/（美）维纽戈帕尔（Venugopal, S.）著. —北京：机械工业出版社，2008.1

（经典原版书库）

书名原文：Data Structures Outside In with Java

ISBN 978-7-111-23165-3

Ⅰ.数… Ⅱ.维… Ⅲ.① 数据结构-英文 ② JAVA语言-程序设计-英文 Ⅳ.TP311.12 TP312

中国版本图书馆CIP数据核字（2007）第205696号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：迟振春

北京京北制版厂印刷 · 新华书店北京发行所发行

2008年1月第1版第1次印刷

170mm × 242mm · 32.25印张

标准书号：ISBN 978-7-111-23165-3

定价：55.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究的范畴，还揭橥了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流逝而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短、从业人员较少的现状下，美国等发达国家在其计算机科学发展的几十年间积淀的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章图文信息有限公司较早意识到“出版要为教育服务”。自1998年开始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与Prentice Hall，Addison-Wesley，McGraw-Hill，Morgan Kaufmann等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出Tanenbaum，Stroustrup，Kernighan，Jim Gray等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及庋藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专程为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近260个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍，为进一步推广与发展打下了坚实的基础。

随着学科建设的初步完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都步入一个新的阶段。为此，华章公司将加大引进教材的力度，除“计算机科学丛书”之外，对影印版的教材，则单独开辟出“经典原版书库”。为了保证这两套丛书的权威性，同时也为了更好地为学校和老师们服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通大学、南京大学、浙江大学、中国科技大学、哈尔滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北

工学院、中国国家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

这两套丛书是响应教育部提出的使用外版教材的号召，为国内高校的计算机及相关专业的教学度身订造的。其中许多教材均已为M. I. T.，Stanford，U.C. Berkeley，C. M. U. 等世界名牌大学所采用。不仅涵盖了程序设计、数据结构、操作系统、计算机体系结构、数据库、编译原理、软件工程、图形学、通信与网络、离散数学等国内大学计算机专业普遍开设的核心课程，而且各具特色——有的出自语言设计者之手、有的历经三十年而不衰、有的已被全世界的几百所高校采用。在这些圆熟通博的名师大作的指引之下，读者必将在计算机科学的宫殿中由登堂而入室。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图书有了质量的保证，但我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。教材的出版只是我们的后续服务的起点。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

电子邮件：hzjsj@hzbook.com
联系电话：(010) 68995264
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

专家指导委员会

（按姓氏笔画顺序）

尤晋元	王　珊	冯博琴	史忠植	史美林
石教英	吕　建	孙玉芳	吴世忠	吴时霖
张立昂	李伟琴	李师贤	李建中	杨冬青
邵维忠	陆丽娜	陆鑫达	陈向群	周伯生
周克定	周傲英	孟小峰	岳丽华	范　明
郑国梁	施伯乐	钟玉琢	唐世渭	袁崇义
高传善	梅　宏	程　旭	程时端	谢希仁
裘宗燕	戴　葵			

Preface

There are two broad approaches to studying data structures.

One is the "inside-out" approach in which the implementation of the data structure, i.e., *how* it is built, is learned either before or in conjunction with its application to problem solving. In other words, start from the nucleus of the data structure and build outward to its use in a problem.

The inside-out approach, however, is discordant with the manner in which software is built in practice out of libraries of objects that are known only through their application programming interfaces (**APIs**). Here, the "outside-in" approach becomes the norm: a component or object is first—and often only—seen via its interface, which characterizes its behavior and therefore its suitability for a given application. That is, *what* a component does precedes how it is built.

Outside-in: From price-tagged interface to implementation

In this book, we are interested in the interface as well as the implementation of data structures. We follow the outside-in approach to presenting them because it will enable students to easily apply in practical software development what they learn in class. Our approach is outlined in the following sequence of steps.

1. **Introduce a data structure by narrating its properties and use in applications.**
   This step familiarizes the student with the characteristic behavior of the data structure, setting the stage for the encapsulation of data and operations into a Java class.
2. **Formalize the characteristic properties of a data structure by presenting the public interface of a Java class that implements the data structure.**
   This step defines the set of operations that may be applied on the data structure, formulated from the discussion in Step 1. With the interface, there also comes a "price tag"—the running times of the interface operations.
   The price tag is an important consideration in the selection of data structures for an application. It may be argued that the price tag determination may only be made after the data structure is implemented. While this is true, the outside-in approach that is used to build software in practice typically separates the group of people who work with the outside from the group of people who build the inside. The outside group must rely on all, and only, that information that comes with the interface. Having a price tag with the interface is critical for the outside group to evaluate and choose the best objects possible for the application at hand.
   To be consistent with this approach, we have attached the price tag to the interface, but made a working compromise: in the interface, specify the minimal requirements of the implementation so that the running times of the operations stay within the price tag.
   We admit that this blurs the separation between interface and implementation, while recognizing that part of the issue is also that the same person—the student—is

working with both the outside and the inside, albeit at different times. The best way for the student to approach the interface-implementation separation is to first imagine that he or she is a client of the data structure, with full cognizance of the interface and the price tag, and then imagine being the implementor of the data structure, who has been told what limitations (price tag) to work with to build the structure.

3. **Further illustrate the use of a data structure by writing Java applications using its class interface presented in Step 2.**

   This step gives the student a clear practical understanding of how to build an application in Java using a data structure whose public interface is known, but whose internal implementation is hidden.

   Steps 2 and 3 strongly emphasize the interface of a data structure. By repeatedly building applications using only the public interface of data structures, the student gets a practical feel for software development using components whose internal implementation details need not (and indeed, may not) be available.

4. **Design and implement the data structure, i.e., develop the code for the Java class whose interface was presented in Step 2, analyze the running times of its operations, and verify them against the price tag.**

   This step emphasizes code reuse in one of two forms: (a) *composition:* using previous data structures as component (Java) objects in building, or composing, a new data structure, or (b) *inheritance:* building a new data structure by inheriting from a previously built data structure (Java class).

   While we are following an outside-in approach, this "in" part does not degenerate into using classes from the Java collections framework. Instead, it is on an equal footing with the "outside" part, with a detailed understanding of the implementation so that the student learns all aspects of building data structures, including evaluating the tradeoffs involved in choosing among a set of candidate implementations.

Apart from providing a consistent pedagogical form, these steps help students to understand and apply the important object-oriented design principles of encapsulation, separation of interface from implementation, and code reuse.

Prerequisites in Java

This book assumes a CS1-level background in Java 1.5, with the following specific coverage: program structure, data types, control structures for decision and repetition, including the `if`, `if-else`, `for`, `while`, and `repeat` statements, and arrays. It also assumes that the student is familiar with the widely used `String` class.

Chapter 1 is a Java primer on object-oriented programming that starts by assuming this CS1-level background, and introduces objects and classes, inheritance, the `Object class`, exceptions, core input and output features, class packaging, and access control. In the context of input/output, the `java.util.StringTokenizer` and the `java.util.Scanner` (new to Java 1.5) classes are described, with a discussion of their typical usage.

The primer also introduces specifically design-oriented features, including polymorphism, abstract classes, and interfaces.

The last section discusses the new Java 1.5 generics, an indispensable tool for building usable and robust data structures. This discussion also details the design and use of the `java.util.ArrayList` class, which is a very useful component in implementing container structures, or collections.

Paths through the book

An essential course focusing on the basic data structures and sorting algorithms, preceded by reviewing/learning the required Java tools and techniques in two weeks, could cover Chapters 1–10, skipping Section 10.7 (AVL Tree), and Chapters 12–13, skipping Sections 13.3 (Heapsort) and 13.4 (Radix sort).

A course that could conduct the Java due diligence in lab instead of lecture could add Heap from Chapter 11, and Heapsort and Radix sort from Sections 13.3 and 13.4 respectively.

Advanced material could be incorporated by covering AVL trees from Section 10.7, and Chapters 14 and 15 on graphs. In case of time limitation, Chapter 14 would suffice to familiarize students with graph algorithms, while leaving out the implementation details of Chapter 15.

A two-course sequence could cover the entire book, including much of the Java background material in Chapter 1. The first course could cover Chapters 1 through 9, up to and including Binary Tree/General Tree, and the second could cover the rest of the chapters, starting with Binary Search Tree/AVL Tree of Chapter 10.

Pedagogical structures.

- Every chapter, except the preliminary Chapters 1 and 2, begins with a list of ***Learning Objectives.*** This gives a precise overview of the learning material in the chapter.
- Key points in every chapter are presented in the following format:
  These key points are also itemized in the end-of-chapter summary.
- Public interfaces of Java classes for data structures are presented as figures in the following format:

Class classname

Constructors

Signature and description of each public constructor

Methods

Running time (price tag), signature, and description of each public method

- Every complete Java class implementation is presented in the following style:

  Class File number *Class File Name*

  Class outline, with some constructors/methods possibly filled in

- Throughout the book, we use algorithms written in pseudo-code, providing language-independent descriptions of processes. These algorithms appear with a header of the form:

  Algorithm name_of_algorithm

  The notations used in the pseudo-code are self-explanatory.
- Every chapter except Chapter 2 concludes with a listing of **Summary** points. These include the key points in the chapter, as well as other important points to remember, including specific Java issues.
- Every chapter except Chapter 2 tests the student's understanding of the material in the form of exercises and programming problems:
  - **Exercises**, which are, for the most part, conceptual language-independent material, especially focusing on work-through reviews, abstract design issues, and time-space analysis.
  - **Programming Problems**, which focus on building Java classes, especially focusing on design/implementation alternatives and code reuse.

Acknowledgments

I would like to thank the Data Structures teams in the Computer Science department at Rutgers University with whom I have taught this course for over more than a decade, and who have provided direct as well as indirect input that has helped shape this book. Special thanks to current and former faculty members Diane Souvaine, Ken Kaplan, Miles Murdocca, and Don Smith for discussing and reviewing the content.

Many thanks to my friends and former graduate student colleagues Nathalie Japkowicz, George Katsaros, and Dan Arena for carefully reviewing the initial C++ draft, Gabriela Hristescu for providing help with typesetting aspects, and Sri Divakaran for reviewing parts of the Java manuscript. Several students in my data structures class of Fall 1997 read the first complete draft of this book and gave valuable feedback. Thanks to them all, especially Alex Chang, for being such a tremendously loyal fan of the book. Thanks also to the students in Fall 2002 and Spring 2003 who pointed out various errors and other shortcomings in an earlier version that was customized for Rutgers.

Thanks are also due to the Data Structures faculty at Middlesex County College, and my former colleagues when I was working at Lucent Technologies with whom I discussed the material in this book at some point or the other. Special thanks to Tom Walsh at Lucent for cheering me on.

Thanks to all the reviewers who helped make this a better book: Barbara Goldner at North Seattle Community College, Mark Llewellyn at University of Central Florida, Chris Dovolis at University of Minnesota, Iyad A. Ajwa at Ashland University, Minseok Kwon at Rochester Institute of Technology, George Rouskas at North Carolina State

University, Roxanne Canosa at Rochester Institute of Technology, Mary Horstman at Western Illinois University, Ray Whitney at University of Maryland, University College, and Robert P. Burton at Brigham Young University.

The editorial team at Prentice Hall have supported me in all aspects of this project. Thanks to my editor Tracy Dunkelberger for believing that this book project would be a worthwhile enterprise, for her cheery confidence that kept me on an even keel, and for making this book real. Thanks to Christianna Lee and Carole Snyder for guiding me through the review process, and for responding to all my questions with patience and grace.

John Shannon, Irwin Zucker, Camille Trentacoste and the production staff at Laserwords helped correct the numerous typographical and grammatical errors that had crept into the book, and formatted the pages to make sure they were presentable. Due to their diligence and expertise, you, dear reader, are spared my inadvertent mistakes or plain ignorance in these areas. For this, my sincere appreciation and thanks to all of them.

I am deeply grateful to the members of my family and that of my wife's for their advice and encouragement, and for being there to pep me up when things didn't seem to be going too well at times. I would specially like to thank my parents, my mother-in-law, and my wife's grandparents for their constant support. Above all, thanks to my father, C.N. Venugopal, and to my wife's grandfather, Dr. Hayrettin Tanacan, for their special participation. Although both of them are far removed from computers in general, and Data Structures in particular, they took the pains to go through my book to try and understand exactly what it was that I was trying to write that was taking so long!

Thanks to my wife, Zehra Tulin, for her help in many ways during the writing of this book, and more importantly, for her love and support at all times. And an extra big hug to my son Amar whose unfailing pride in his dad spurs me on.

Finally, thanks to all the students who took my classes in various subjects at Rutgers University and Middlesex County College over the years. They have been instrumental in my growth as an educator, and I hope this book can serve as a token of my gratitude to each and every one of them.

In closing, I would be very happy to hear from you about this book—what you liked, what you did not care for, and the errors you found, if any. You can reach me by email at *sesh_venugopal@rutgers.edu*. Your feedback will help me serve you better.

Sesh Venugopal

Piscataway, New Jersey
November, 2006

List of Class Files

Contents